IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 4, August 2026, pp. 3836~3852
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i4.pp3836-3852

Experimental validation of vision-based hybrid navigation for differential-drive robots

Thanh Quang Duc Le[1], Trong Hieu Luu[2]
[1]Faculty of Electrical and Electronics Engineering, Vinh Long University of Technology Education, Vinh Long, Vietnam
[2]College of Engineering, Can Tho University, Can Tho, Vietnam

Article Info

Article history:

Received Nov 11, 2025
Revised Jul 11, 2026
Accepted Jul 21, 2026

Keywords:

Autonomous mobile robot
Dynamic window approach
Particle swarm optimization
Sensor fusion system
Vision-based navigation system

ABSTRACT

Reliable mobile-robot deployment requires mapping, localization, global planning, and real-time obstacle avoidance to operate consistently under sensor noise and physical constraints. This study presents a vision-based hybrid navigation framework for a differential-drive mobile robot in a warehouse-like environment. The main contribution lies in the system-level integration and experimental validation of established techniques rather than the development of a new standalone navigation algorithm. A ceiling-mounted camera converts top-view images into an occupancy-grid map and world coordinates, while a convolutional neural network (CNN) recognizes goal markers. Odometry–augmented reality University of Cordoba (ArUco) fusion is used to correct accumulated localization drift. Particle swarm optimization (PSO) generates smooth global paths offline, whereas the dynamic window approach (DWA) performs real-time local motion control. Experiments in a 3.3 m × 2.4 m workspace achieved average obstacle-localization errors of 0.959 cm and 0.696 cm along the x- and y-axes, respectively, and a goal-recognition accuracy of 99%. The DWA controller required 12.61±2.42 ms per cycle, while rapidly-exploring random tree (RRT) and PSO required 4–7 s and 477–692 s, respectively, for global path generation. Simulation and physical experiments confirmed collision-free navigation and successful quick response (QR)-code-based goods inspection, demonstrating the feasibility of the proposed framework for small, structured indoor environments.

Corresponding Author:

Trong Hieu Luu
College of Engineering, Can Tho University
3/2 street, Ninh Kieu Ward, Can Tho City, Vietnam
Email: luutronghieu@ctu.edu.vn

1. INTRODUCTION

Path planning is a core task in autonomous robot navigation, where a collision-free route must be generated from a start point to a target in a cluttered environment. Existing approaches include heuristic search, intelligent optimization, and reinforcement learning, and each method involves trade-offs in computation time, adaptability, and path quality. Among classical methods, $A*$ is widely used because it combines graph search with heuristic guidance, providing efficient shortest-path search and practical performance in static maps. Previous studies reported fast search speed [1]–[3], high success rate [4], and improved path smoothing performance [5] for $A*$ in structured environments, but its effectiveness can decrease in large-scale or dynamic scenes that require frequent replanning.

To improve flexibility, optimization-based algorithms such as artificial potential field (APF), genetic algorithm (GA), and ant colony optimization (ACO) have been investigated. APF is computationally

light and responds quickly [6], [7], and good success rates have been reported through proper parameter tuning [8]–[10], but local minima may occur in complex obstacle configurations [11]. GA can explore a wide solution space and produce near-optimal paths [12], [13] with dynamic obstacle avoidance capability [14], [15], although convergence can be slow or premature when population diversity is insufficient. ACO can achieve high search efficiency [16] and generate smooth paths [17]–[19], but it may suffer from suboptimal convergence [20] and increased computational cost in large problems due to pheromone updates. Reinforcement learning has also been applied to navigation, where Q-learning can learn policies through interaction without a complete model and may converge under certain conditions [21]–[23]; however, slow convergence [24], sensitivity to local optima [25], and reduced efficiency in high-dimensional spaces remain concerns. Monte Carlo-based methods, such as Monte Carlo tree search (MCTS) and its variants, can improve global search and convergence properties [26]–[29], but they still require extensive computation and may be limited in strict real-time settings.

Although the algorithms have achieved remarkable progress in navigation research, most of them are still validated only through simulation or benchmark challenge problems. In practice, mobile robots operating in real-world environments—such as warehouses or factories—must deal with uncertain perception, sensor noise, and dynamic obstacles. Under such conditions, traditional or purely simulation-based algorithms often face difficulties in maintaining smooth motion, stable localization, and adaptive goal recognition. This gap highlights the necessity for a more integrated and experimentally validated navigation framework that combines the strengths of optimization, perception, and control.

To address the gap between theoretical path planning and practical deployment, this study introduces an integrated vision-based navigation framework. The system integrates global planning via particle swarm optimization (PSO) and real-time local control using the dynamic window approach (DWA). The main contribution of this work lies in the system-level integration and experimental validation of established navigation techniques rather than the development of a fundamentally new algorithm. Specifically, the proposed framework as follows:

- Converts ceiling-camera images into occupancy-grid maps and world coordinates while using convolutional neural network (CNN)-based recognition to identify navigation targets.
- Combines offline PSO global planning, real-time DWA local control, and odometry–augmented reality University of Cordoba (ArUco) fusion for smooth navigation and drift correction.
- Validates the complete system through simulation and real-world experiments involving autonomous navigation, goods inspection, and quick response (QR)-code recognition in warehouse-like environment.

2. METHOD

2.1. System overview

The intelligent control algorithm is designed to perform high-level decision-making and trajectory generation for the mobile robot, as in Figure 1. In this study, a PSO algorithm is adopted to determine the optimal robot behavior and navigation path in the warehouse environment. The PSO module processes global information to plan collision-free and efficient trajectories, which are then transmitted wirelessly to the robot platform for execution. Additionally, a ceiling-mounted camera is utilized to monitor the robot's position in real time, providing feedback on global localization and ensuring that the robot accurately follows the desired trajectory within the operational area.

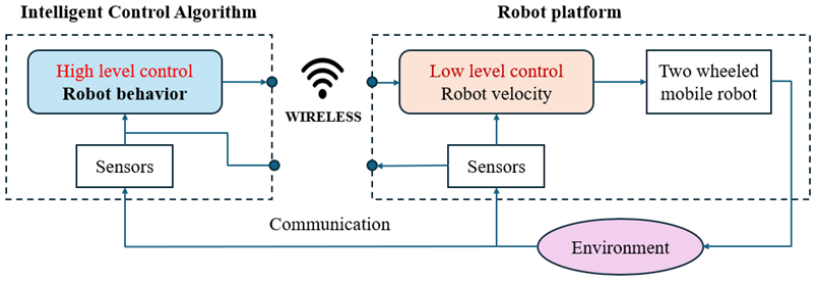


Figure 1. Intelligent control framework for the two-wheeled differential-drive mobile robot

The robot platform is responsible for low-level motion control and real-time operation of the differential-drive mobile robot. A proportional–integral–derivative (PID) controller regulates the wheel velocities to achieve smooth and stable motion based on the reference trajectory received from the high-level

controller. The robot integrates multiple onboard sensors, including a light detection and ranging (LiDAR) sensor for obstacle detection and avoidance, and wheel encoders for velocity and displacement feedback. These sensor data are continuously exchanged with the high-level controller to update the environment map and adjust the robot's movement dynamically. This closed-loop control structure ensures accurate velocity tracking, effective obstacle avoidance, and safe autonomous navigation in warehouse environments for automated goods transportation.

2.2. Velocity control and pose estimation of the mobile robot

2.2.1. Kinematic model of the differential-drive mobile robot

The mobile robot used in this study is a differential-drive type equipped with two independently driven wheels separated by a fixed distance L, where O the midpoint between the wheels. The linear velocity of the left wheel and right wheel are v_L and v_R, respectively. Robot moves forward, backward and rotates around its instantaneous center of curvature (ICC) as shown in Figure 2.

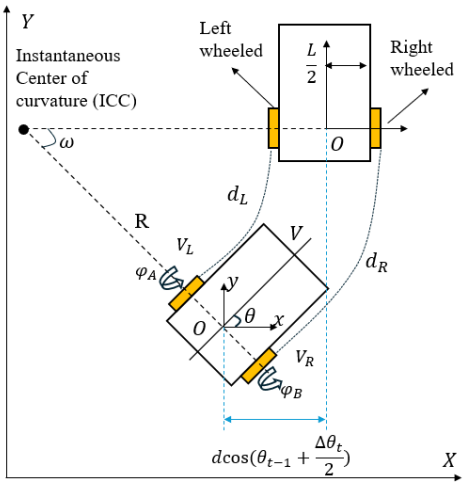


Figure 2. Kinematic and odometry model of a differential-drive mobile robot

The robot linear and angular velocity are calculated by (1) and (2).

$$v = \frac{v_R + v_L}{2} \tag{1}$$

$$\omega = \frac{v_R - v_L}{L} \tag{2}$$

The robot pose in the global frame is described by the vector $O = [x, y, \theta]^T$, where (x, y) represents the position of point O and θ the heading angle. The robot's kinematic model can be expressed as (3).

$$\begin{cases} \dot{x} = v cos\theta \\ \dot{y} = v sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{3}$$

Over a sampling Δt, mobile robot in the global frame can be described as (4).

$$\begin{cases} x_{t+1} = x_t + v_t cos\Delta t \\ y_{t+1} = y_t + v_t sin\Delta t \\ \theta_{t+1} = \theta_t + \omega\Delta t \end{cases} \tag{4}$$

2.2.2. Velocity control using proportional–integral–derivative

To maintain accurate and stable motion, a PID controller is implemented for each wheel motor. Encoder feedback provides the actual wheel velocities v_L and v_R, while the controller ensures that they follow the reference velocities v_L^{ref} and v_R^{ref}. The discrete PID control law is calculated as (5).

$$v_q[k] = K_P e_q[k] + K_I \Delta t \sum_{i=0}^{t} e_q[j] + K_D \frac{e_q[k] - e_q[k-1]}{\Delta t} \tag{5}$$

Where $v_q[k]$ is the desired velocity at sample k; $e_q[k] = v_q^{ref}[k] - v_q[k]$ is the velocity error at sample k; $q \in \{L, R\}$ is left or right wheel; and K_P, K_I, K_D are proportional, integral, and derivative gains, respectively.

The PID output $v_q[k]$ generates the motor command signal (e.g., PWM duty cycle). Both wheel controllers operate simultaneously to achieve smooth linear velocity $v$and angular velocity ω, which are then used by the DWA for real-time trajectory tracking. To ensure feasible motion, the following dynamic constraints are calculated (6).

$$\begin{cases} |v - v_{current}| \leq \alpha_{(v)max} \Delta t \\ |\omega - \omega_{current}| \leq \alpha_{(\omega)max} \Delta t \end{cases} \quad (6)$$

Where $a_{(v)max}$ and $\alpha_{(\omega)max}$ are the robot's maximum linear and angular accelerations.

2.2.3. Odometry-based pose estimation

The robot's pose is estimated from wheel encoder readings using odometry. The incremental displacements of the left and right wheels in a sampling interval are Δd_L and Δd_R. The robot's motion increments are calculated as (7) and (8).

$$\Delta d = \frac{\Delta d_L + \Delta d_R}{2} \quad (7)$$

$$\Delta\theta = \frac{\Delta d_R - \Delta d_L}{L} \quad (8)$$

The updated pose is given by (9).

$$\begin{cases} x_{t+1} = x_t + \Delta d \cdot cos(\theta_t + \frac{\Delta\theta_t}{2}) \\ y_{t+1} = y_t + \Delta d \cdot sin(\theta_t + \frac{\Delta\theta_t}{2}) \\ \theta_{t+1} = \theta_t + \Delta\theta \end{cases} \quad (9)$$

2.3. Mobile robot path planning

2.3.1. Global path planning using particle swarm optimization

Global path planning is an essential problem in mobile robot navigation, especially in complex environments that contain multiple obstacles. In this study, the PSO [30] algorithm is applied to generate an optimal path from the starting point to the target point while ensuring that the robot avoids collisions with obstacles during motion. The global path planning process can be described as finding a set P of discrete points generated by the robot as it moves through the environment, where each consecutive segment between points does not intersect any static obstacle. The set of points P can be represented as (10).

$$P = \{S, p_1, p_2, \dots, p_n, T\} \quad (10)$$

The total path length is calculated as (11).

$$L = \sum_{i=1}^{n-1} d(i, i+1) \quad (11)$$

Where S is the start point; T is the target point; $(p_1, p_2, \dots, p_n)$ is the sequence of intermediate points in the environment; $d(i,\ i+1)$ is the distance between consecutive points; p_i and p_{i+1}, and n is the total number of points in the set P.

To make the global path smoother and more suitable for a mobile robot, a cubic spline interpolation method is applied to optimize the path. This process reduces sharp turns and creates a continuous, smooth curve. The procedure for generating the cubic spline path between two points p_i and p_{i+1} is as follows:

i) Generate n interpolated points between each pair of points p_i and p_{i+1} using cubic spline interpolation.
ii) Connect all interpolated points $(p_1, p_2, \dots, p_n)$ to obtain a continuous cubic spline path.
iii) The local spline length $C(i, i+1)$ between two points p_i and $p_{i+1}$is is calculated as (12).

$$C(i, i+1) = \sum_{k=1}^{m-1} \Delta s_k = \sum_{k=1}^{m-1} \sqrt{(\Delta x_k)^2 + (\Delta y_k)^2} \quad (12)$$

Finally, the total cubic spline path length L can be expressed as (13). This cubic spline-based optimization ensures that the global path is smooth, continuous, and feasible for real-time navigation of the differential-drive mobile robot.

$$L = \sum_{i=1}^{n-1} \mathrm{C}(i, i+1) \quad (13)$$

In the proposed system, the PSO algorithm searches for the optimal global path represented by the set P. Each particle in the swarm encodes a potential trajectory composed of the coordinates of $(p_1, p_2, \dots, p_n)$. The search space corresponds to the free-space cells of the environment map, where 0 represents traversable areas and 1 denotes obstacles. Each particle i has a position x_i^t and velocity v_i^t at iteration t. The quality of each path is evaluated using a fitness function that combines the total path length and an obstacle penalty term in (14).

$$F_i = L_i + \lambda \cdot N_{obs} \tag{14}$$

Where L_i is calculated from (10) and (11), N_{obs} is the number of path segments intersecting obstacle cells, and λ is a penalty coefficient that discourages unsafe paths. The velocity and position of each particle are updated according to the standard PSO in (15) and (16).

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 r_1 \left(p_{i,best} - x_i^t\right) + c_2 r_2 (g_{best} - x_i^t) \tag{15}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{16}$$

Where ω is the inertia weight; $c_1$and c_2 are cognitive and social learning coefficients; $r_1, r_2 \in [0,1]$ are random variables introducing stochastic exploration; and $p_{i,best}$ and g_{best} represent the personal and global best positions, respectively.

If an updated waypoint p_i lies within an obstacle region, the algorithm modifies its direction using either a reflection strategy or an adaptive penalty adjustment, forcing the particle to explore valid regions. The iterative process continues until one of the following conditions is satisfied:

– The maximum number of iterations is reached.
– The improvement in F_i between iterations is negligible.
– The swarm converges to a stable global solution.

Upon convergence, the best path obtained from the swarm is selected as the global optimal path. To improve reproducibility, the PSO hyperparameter settings used for global path planning are summarized in Table 1. The swarm size was set to $N = 1{,}000$, the inertia weight was fixed $\omega = 1.0$, and the cognitive and social learning coefficients were set to $c_1 = 1.5$ and $c_2 = 1.5$, respectively. The optimization was terminated when the maximum iteration limit $i_{max} = 400$ was reached, or earlier if convergence was observed. These settings provide a practical balance between solution quality and computational cost for the indoor workspace considered.

Table 1. PSO parameter settings for global path planning

Parameter	Symbol	Recommend value	Role
Swarm size	N	1000	Number of particles in PSO
Inertia weight	ω	1.0	Controls the influence of previous velocity
Cognitive coefficient	c_1	1.5	Weight of learning from the particle's best position
Social coefficient	c_2	1.5	Weight of learning from the global best particle
Maximum iterations	i_{max}	400	Maximum number of iterations

2.3.2. Local path planning using dynamic window approach

The motion of a differential-drive mobile robot can be expressed by the following kinematic equations [31]. The DWA algorithm searches for the best pair (v, ω) considering the robot's kinematics and dynamic limits ((5) and (6)). The feasible window is defined by (17).

$$\begin{cases} v_{min} \le v \le v_{max}, \\ \omega_{min} \le \omega \le \omega_{max} \\ |v - v_{current}| \le \alpha_{max} \Delta t, \\ |\omega - \omega_{current}| \le \alpha_{max} \Delta t \end{cases} \tag{17}$$

And safety constraints in (18).

$$d(v, \omega) \ge d_{safe} \tag{18}$$

Where $d(v, \omega)$ is the minimum distance from the robot to the nearest obstacle and d_{safe} is the minimum safety distance. Each pair (v, ω) creates a short, predicted trajectory, which is evaluated by a composite objective function (19).

$$G(v,\omega) = \alpha \cdot heading(v,\omega) + \beta \cdot distance(v,\omega) + \gamma \cdot velocity(v,\omega) \quad (19)$$

Where $Heading(v,\omega)$ measures the alignment between the robot's heading and the direction to the target; $Distance(v,\omega)$ evaluates the minimum distance to surrounding obstacles; $Velocity(v,\omega)$ represents the magnitude of forward speed; and α, β, γ are weighting coefficients that balance these criteria.

The algorithm evaluates all feasible (v,ω) pairs within the dynamic window and selects the pair with the highest $G(v,\omega)$. The corresponding trajectory is then executed by the robot during the next control interval. This process is continuously repeated, enabling the robot to adapt its motion in real time. The DWA parameters used for real-time local motion control are summarized in Table 2. The velocity and acceleration limits were selected according to the physical capability of differential-drive robot, while objective-function weights were empirically tuned to balance target alignment, obstacle clearance, and forward motion.

Table 2. DWA parameter settings

Parameter	Symbol	Value
Maximum linear velocity	v_{max}	0.15 m/s
Maximum angular velocity	ω_{max}	0.8 rad/s
Maximum linear acceleration	a_{max}	0.2 m/s^2
Prediction horizon	T_p	2 s
Minimum safety distance	d_{safe}	0.1 m
Heading weight	α	0.4
Clearance weight	β	0.4
Velocity weight	γ	0.2

2.4. Experiment setup

2.4.1. Environment setup and hardware system

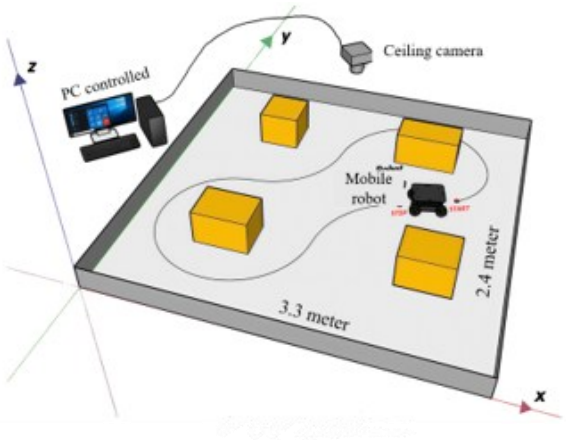
The experimental setup is illustrated in Figure 3, which simulates the movement and self-localization of a two-wheeled differential-drive mobile robot operating in an environment with multiple obstacles. The experiment was conducted on an experimental area with a total area of 3.3 m × 2.4 m, where obstacles were randomly placed within the workspace (Figure 3(a)). The mobile robot started from position 0 and was programmed to follow a trajectory passing through all numbered targets in sequence 0 → 1 → 2 → 3 → 4 → 5 → 0 (Figure 3(b)). Each target position was detected and recorded using an overhead ceiling camera, which provided global coordinate references during the experiment.

The robot used in this study is shown in Figure 3(c), and its physical dimensions are listed in Table 3. The robot is equipped with an Intel RealSense red, green, blue–depth (RGB-D) camera mounted at the front to capture visual information, and two-wheel encoders for odometry computation. The LattePanda embedded computer serves as the main controller, integrated with a microcontroller responsible for driving the direct current (DC) motors, reading sensor data, and communicating with the host computer.

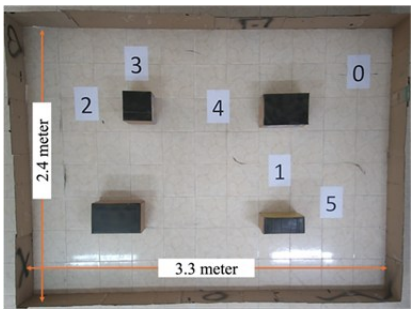
To estimate the robot position accurately within the workspace, an ArUco marker is mounted on the top of the robot. The ceiling camera is used for sensor fusion by detecting robot positions and evaluating data with robot's pose odometry. It continuously detects this marker to determine the robot's global coordinates in real time. The position obtained from the ceiling camera is then fused with the incremental motion data from the wheel encoders to update the robot pose on the working map. This hybrid localization approach combines vision-based global positioning and odometry-based local estimation, effectively reducing accumulated drift and improving navigation accuracy during operation.

2.4.2. Environment modeling and image-to-world coordinate conversion

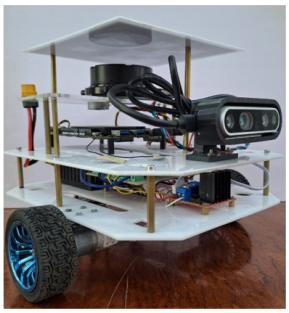
The workspace environment is modeled through an image-processing pipeline that extracts geometric features from the ceiling camera and converts image coordinates into real-world measurements. The captured image is first cropped into the region of interest and converted from the RGB to the HSV color space. Upper and lower HSV thresholds $[H_{min}, S_{min}, V_{min}]$ and $[H_{max}, S_{max}, V_{max}]$ is then applied to generate a binary mask that highlights obstacle surfaces. Figure 4 shows the resulting binary masks generated by this pipeline for two different camera views used in obstacle detection: the surface floor view and the side floor view. As shown in Figure 4(a), the initial mask may contain noise and small gaps along object boundaries. Therefore, Gaussian blurring is used to suppress high-frequency noise, and morphological closing is applied to fill small holes and connect broken contours, producing a cleaner segmentation result for subsequent contour extraction and map generation, as illustrated in Figure 4(b).



(a)



(b)

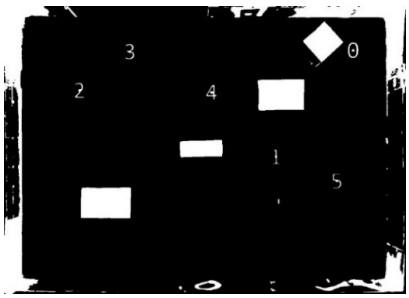


(c)

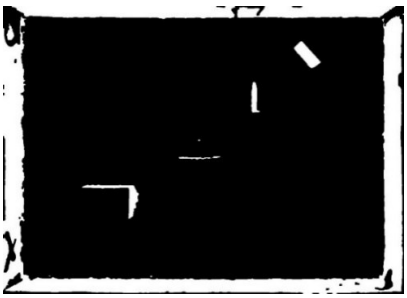
Figure 3. System layout and experimental space: (a) system layout, (b) the experimental space, and (c) two-wheeled differential mobile robot

Table 3. Physical parameters of the differential-drive mobile robot

Parameters	Dimension (m)
Size	0.2 × 0.15 × 0.2
Wheelbase	0.25
Wheel radius	0.0325



(a)



(b)

Figure 4. Image processing pipeline to detect obstacles (a) binary image of the surface floor and (b) binary image of the side view of the floor

Contours C_i are extracted and filtered based on the area condition $A_{min} < A(C_i) < A_{max}$ to eliminate non-relevant objects. Each valid contour is approximate to a polygon, and quadrilateral contours are retained as candidate obstacle boundaries. The four corner points of each obstacle are then converted from image coordinates (u, v) to metric coordinates (x_α, y_α) by a linear scaling model in (20).

$$\begin{cases} x_\alpha = a_x(u - u) \\ y_\alpha = a_y(v_0 - v) \end{cases} \quad (20)$$

Where (u_0, v_0) are the image center, and α_x, α_y are the conversion factors (m/pixel) along the x-axes and y-axes, respectively. The coordinate system is later shifted so that the origin is relocated to the lower-left corner of the workspace to match the robot's global map coordinate frame as shown in Figure 5 (21).

$$\begin{cases} x_\beta = x_\alpha + \Delta x \\ y_\beta = y_\alpha + \Delta y \end{cases} \quad (21)$$

Where $(\Delta x, \Delta y)$ are translation offsets between the image and workspace origins.

The resulting environment is represented as a binary occupancy grid with dimensions 240 × 330, in which each cell corresponds to 1 cm^2 of real space. This grid encodes both the boundary walls and the detected static obstacles. To determine the robot's position, an ArUco marker is mounted on its top surface. The centroid of the detected marker in image coordinates (u_A, v_A) is converted into real-world coordinates (x_A, y_A) by (20). The robot's orientation θ is estimated from the direction vector between the top-left and top-right corners of the marker (Figure 6) (22).

$$\theta = tan^{-1}\left(\frac{y_R - y_L}{x_R - x_L}\right) \quad (22)$$

Where (x_L, y_L) and (x_R, y_R) are the real-world coordinates of the two corners. The angle θ is normalized within the range $[0, 2\pi]$, counterclockwise in the upper semicircle and clockwise otherwise.

For identifying goal positions, digit labels attached to the floor are detected using a ConvNet deep learning [32] trained on the MNIST dataset. The dataset and training configuration are summarized in Table 4. The centroid of each recognized digit is transformed into real-world coordinates using (20), defining the corresponding target locations in the workspace.

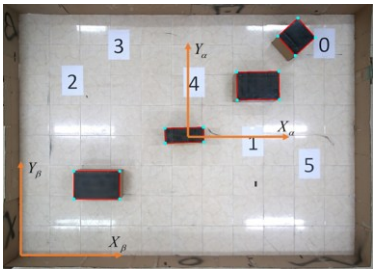


Figure 5. Geometric relationship between the image coordinate system and real-world map coordinates

Figure 6. Localization principle using ArUco marker and orientation estimation

Table 4. Dataset training and ConvNet training configuration

Dataset	Parameters
Input image size	28 × 28 pixel
Number of training images	60,000
Number of testing images	10,000
Batch size	8
Maximum number of epochs	12

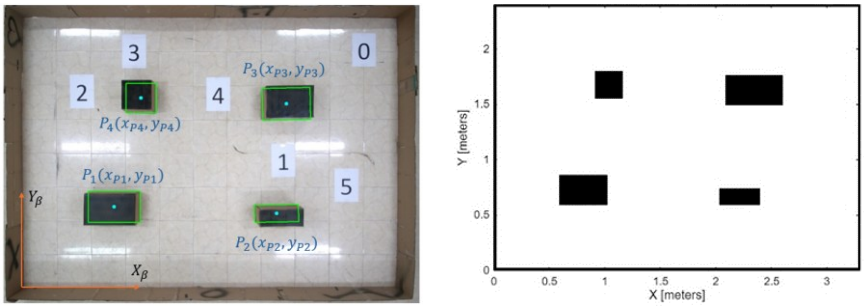
3. RESULTS AND DISCUSSION

3.1. Vision-based perception and mapping results

3.1.1. Obstacle detection and occupancy-grid generation

The environmental map was constructed from top-view images captured by the ceiling camera and processed in MATLAB. After applying image thresholding and segmentation, the obstacle and free-space regions were clearly distinguished. Figure 7 illustrates the resulting digital map, where the identified obstacles are represented in distinct color regions to facilitate robot navigation and environment modeling. Figure 7(a) shows the overhead camera view used for obstacle detection and coordinate translation. Obstacles are detected and enclosed by bounding boxes, and their reference points are expressed in the robot base frame

(X_B, Y_B) as $P_1(x_{p1}, y_{p1})$ to $P_4(x_{p4}, y_{p4})$. The numbered markers indicate goal/location IDs used for navigation. Figure 7(b) presents the generated digital map in metric coordinates. The detected obstacles from the camera image are converted into an occupancy representation, where black regions denote occupied cells and white regions denote free space, providing a planning map for global and local navigation.



(a) (b)

Figure 7. Goal position, object detection, and coordinate translation: (a) obstacle detection and (b) digital map generation

Quantitative results of obstacle localization accuracy are presented in Table 5. The average absolute errors in the x- and y-coordinates are 0.959 cm and 0.696 cm, respectively, indicating that the vision-based detection system achieves high localization precision. These results demonstrate the robustness of the proposed image-processing pipeline and the coordinate transformation model, which enable reliable obstacle identification and mapping within the real-world workspace.

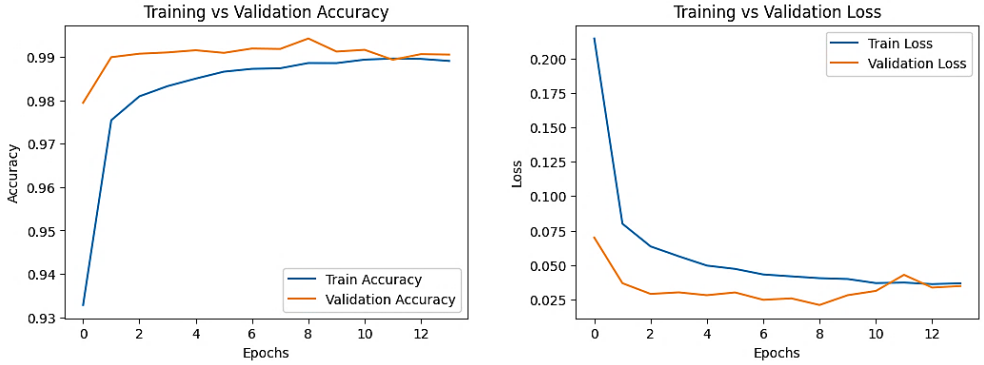
Table 5. Obstacle location detection results

Obstacles	Real coordinate (X_β, Y_β)		Image coordinate $(P_i(x_{Pi}, y_{pi})$		Error	
Obstacle 1	204.50	75.00	204.57	74.68	0.07	-0.32
$P_1(X_{P1}, Y_{P1})$	240.00	75.00	243.51	75.31	3.51	0.31
	240.00	60.00	243.82	60.31	3.82	0.31
	204.50	60.00	204.89	59.68	0.39	-0.32
Obstacle 2	60.00	86.50	57.76	87.44	-2.24	0.94
$P_2(X_{P2}, Y_{P2})$	102.00	86.50	103.08	87.44	1.80	0.94
	102.00	60.00	103.08	59.68	1.08	-0.32
	60.00	60.00	57.76	59.68	-2.24	-0.32
Obstacle 3	210.00	177.00	211.27	178.40	1.27	1.40
$P_3(X_{P3}, Y_{P3})$	251.00	177.00	253.40	179.04	2.40	2.04
	251.00	150.00	254.04	150.95	3.04	0.95
	210.00	150.00	211.59	150.31	1.59	0.31
Obstacle 4	92.00	180.00	90.63	182.23	-1.37	2.23
$P_4(X_{P4}, Y_{P4})$	116.00	180.00	116.80	182.23	0.80	2.23
	116.00	156.00	116.80	156.38	0.80	0.38
	92.00	156.00	90.63	156.38	0.63	0.38
		Average error			0.959	0.696

3.1.2. Convolutional neural network -based goal-marker recognition

A CNN ConvNet [32] was trained to recognize digit markers serving as goal identifiers for the mobile robot. To improve generalization, data augmentation techniques were applied during training. The training configuration is summarized in Table 4, with a batch size of 8, 12 epochs, and early stopping enabled to prevent overfitting. As shown in Figure 8, both training and validation accuracies increased rapidly, reaching approximately 99% after the fourth epoch, while the corresponding loss curves decreased and converged smoothly. This behavior indicates that the model learned efficiently without signs of overfitting. Evaluation metrics listed in Table 6 demonstrate consistent results across all classes, with average precision = 0.99, recall = 0.99, and F1-score = 0.99 (Figure 8(a)). The uniformity among classes, with several achieving perfect scores (1.00), confirms that the CNN model is well-balanced and unbiased. With an overall accuracy of 99% on 10,000 test images, the trained model proves effective and reliable for practical deployment in goal detection tasks. In Figure 8(b), the training loss and validation loss over epochs. Both curves decrease sharply during the first few epochs and then decline more gradually, indicating stable convergence. The

validation loss remains low and relatively stable, with a small increase around epochs 10–11, which may suggest mild overfitting; however, the overall gap between the two curves stays small, implying that the model maintains good generalization performance. Figure 9 illustrates the final localization results, confirming that the proposed CNN-based approach can accurately determine the positions of target points with negligible errors—sufficient for precise path planning and robot navigation in static environments.



(a) (b)

Figure 8. Model training and validation of (a) accuracy and (b) loss

Table 6. The CNN model performance

	Predicted	Recall	F1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	0.99	0.99	0.99
3	0.99	1.00	0.99
4	0.99	0.99	0.99
5	1.00	0.99	0.99
6	1.00	1.00	1.00
7	0.99	0.99	0.99
8	0.99	1.00	1.00
9	0.99	0.99	0.99
Average weighted	0.99	0.99	0.99

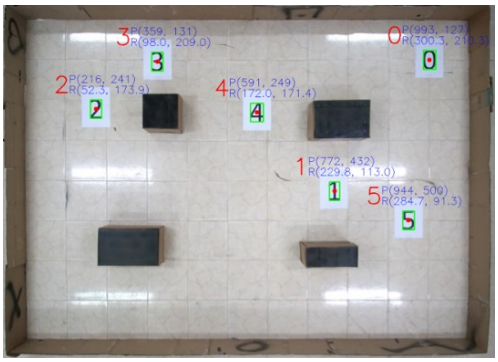


Figure 9. Position detection

3.2. Localization using odometry and ArUco correction

Two localization methods were tested: i) using odometry from wheel encoders and ii) combining odometry with ArUco-marker detection. Figure 10 compares the trajectories estimated by the two localization methods with the real (ground-truth) trajectory. This comparison quantifies the localization accuracy improvement achieved through ArUco-marker correction. In detail, Figure 10(a) shows the trajectory estimated by odometry alone. At first, the path matches the real trajectory, but errors gradually increase due to wheel slip and encoder noise. Figure 10(b) shows the result of the combined method. When the robot detects ArUco markers, its global position is corrected, which removes the drift from odometry. The combined approach achieves smaller errors, around 0.01–0.05 m, and provides more stable localization.

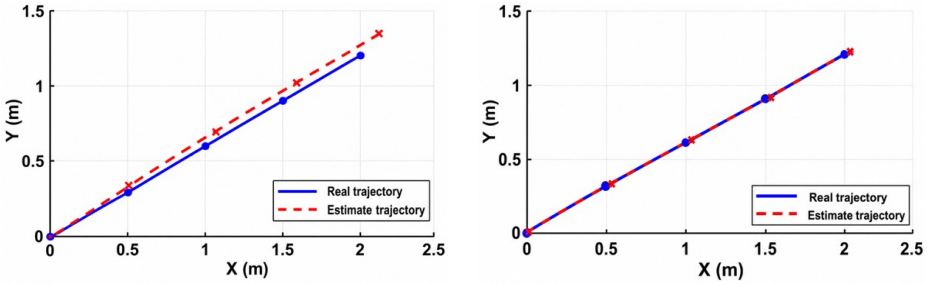


Figure 10. Mobile robot odometry using sensor fusion system (a) mobile robot trajectory using wheel encoders and (b) mobile robot trajectory by sensor fusion

3.3. Simulation results and comparative benchmarking

The proposed navigation framework, which integrates global path planning and ArUco-based localization, was first verified through MATLAB simulation. Figure 11 presents the simulated trajectories obtained under two operating scenarios: standard path planning and path planning combined with a goods-transport task. This comparison evaluates the robot's path-planning and obstacle-avoidance performance under both conditions. In detail, Figure 11(a) shows the simulated trajectory of the mobile robot. The robot successfully followed the pre-planned route while avoiding all obstacles in the environment. The motion path was smooth, and the robot adjusted its steering effectively when approaching obstacle boundaries. To evaluate the applicability in warehouse-like conditions, an additional simulation was performed where the robot transported goods between several predefined locations (Figure 11(b)). The robot was able to move between shelves and corridors with accurate trajectory tracking and no collisions.

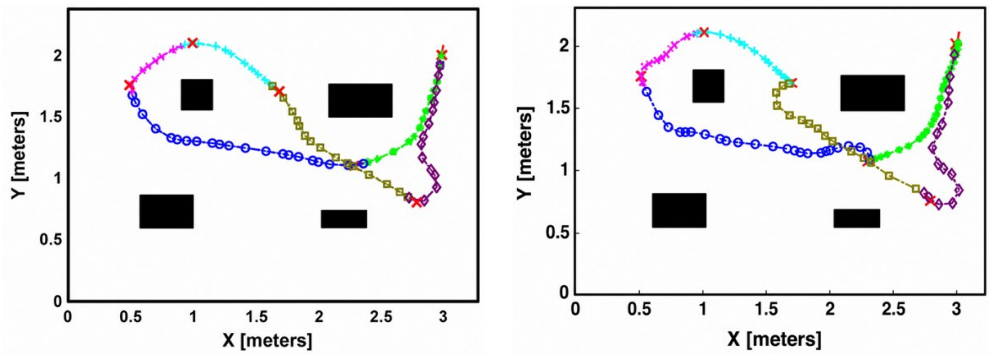


Figure 11. Mobile robot path planning and inspection task simulation: (a) path planning simulation and (b) path planning with goods inspection task

Computational efficiency was evaluated to assess the real-time feasibility of the proposed motion planning and control framework. All simulations were executed on a PC equipped with an Intel Core i5-14400F CPU and 32 GB RAM. In this framework, PSO is used only for global path generation in simulation and is executed outside the real-time control loop. During navigation, local motion control is performed by the DWA planner at every control step using real-time observations from the ceiling-mounted camera and onboard sensing. The timing results are summarized in Table 7. The end-to-end loop achieved an average cycle time of 17.35±5.18 ms, with a maximum of 215.13 ms, corresponding to an average loop frequency of 57.64 Hz. The DWA module required 12.61±2.42 ms on average, with a maximum of 29.30 ms, supporting real-time collision-aware control. In real-world trials, one full navigation cycle across the workspace was completed in approximately 180 s (about 3 min), indicating consistent behavior between simulation and physical experiments under the same ceiling-vision setup. Occasional worst-case spikes were observed and were mainly caused by visualization and logging overhead rather than the core DWA computation.

Table 7. Timing analysis of global planning and real-time control for the proposed navigation system

Module/Metric	Scenario	Min	Avg + std	Max	Unit
PSO global planning time	Simulation only	477	—	692	second
DWA cycle time	Simulation only	—	12.61±2.42	29.30	ms
Overall loop time	Simulation only	—	17.35±5.18	215.13	ms
Average loop frequency	Simulation only	—	57.64	—	Hz
Full navigation time	Real-world	—	~180	—	second

A comparative simulation study was conducted under the same workspace map and identical start–goal conditions to benchmark the proposed navigation strategy. Figure 12 compares the simulated navigation results of the three baseline methods. Figure 12(a) presents the rapidly-exploring random tree (RRT-generated path); Figure 12(b) shows the path obtained using standalone PSO, and Figure 12(c) illustrates DWA-only navigation without global PSO guidance. The PSO-based planner produced the most optimized trajectory among the compared methods, and a smoother route was obtained for differential-drive motion. In contrast, the RRT planner generated a feasible collision-free path rapidly, but sharper turns may appear due to its sampling-based nature. For the DWA-only case, a trajectory trend similar to PSO can be observed in open regions; however, less optimal behavior is typically observed near turning corners because decisions are made locally within a limited dynamic window, which may lead to wider turns or short detours when navigating around corner-like structures.

In addition, simulation runtime was reported to clarify the efficiency of the trade-off among the compared methods (Table 8). RRT achieved the fastest global planning time (from 4 to 7 s), whereas PSO required a longer computation time (about 477–692 s per planning run) to converge to an optimized global solution in the current implementation. The DWA-only baseline completed the navigation task in approximately 35 s. These results indicate that RRT is suitable for fast feasible planning, whereas PSO is more computationally demanding but provides higher-quality global paths, and DWA can operate in real time but may lose optimality around sharp corners without global guidance.

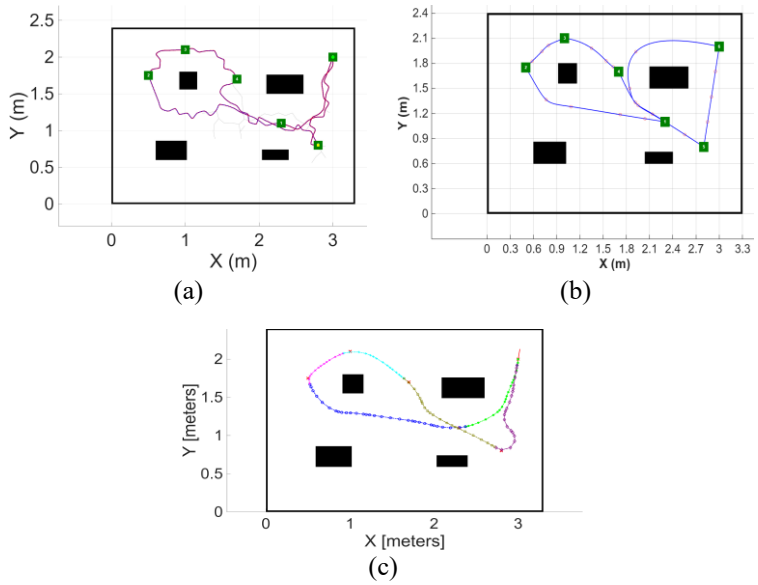


Figure 12. Comparative simulated trajectories for global and local planning baselines: (a) RRT-based global planning, (b) pure PSO-based global planning, and (c) DWA-only navigation without PSO guidance

Table 8. Runtime comparison of baseline planners in MATLAB simulation

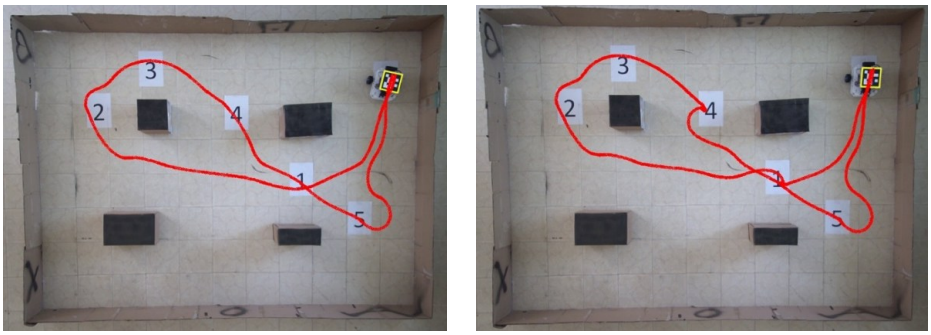
Method/Metric	Scenario	Min	Avg	Max	Unit
RRT global planning time	Global path generation	4	—	7	second
PSO global planning time	Global path generation	477	—	692	second
DWA-only navigation time	Global path generation	35	—	35	second

3.4. Real-world navigation and goods-inspection experiments

The real-world experiments were performed to evaluate the robot's navigation performance and goods-inspection ability in a warehouse-like environment. Figure 13 presents the robot's real-world

trajectories recorded under the two experimental scenarios, comparing global path planning alone against global path planning combined with the goods-inspection task. In detail, Figure 13(a) shows the actual trajectory of the robot when operating with only global path planning, while Figure 13(b) illustrates the case when the robot performed global path planning combined with the goods-inspection task. At each numbered position, the robot stopped and rotated once to scan for QR-codes placed on the storage shelves. During operation, the robot's trajectory was updated in real time. The red line in Figure 13 represents the real-time path generated by the sensor-fusion system that combines odometry and ArUco-marker localization. The recorded trajectory shows that the robot accurately followed the planned path, confirming the stability and precision of the sensor-fusion-based localization method.

Figure 14 shows representative QR-code recognition results obtained during the automated goods-inspection task. At each predefined target, the robot stopped and scanned the QR code attached to the corresponding shelf. The decoded information was then used to identify the product and its storage location, confirming that the proposed system could integrate navigation and visual inspection within a single operational cycle.



(a) (b)

Figure 13. Mobile robot path planning and inspection task in real-world experiments (a) mobile robot motion planning in real-time control and (b) mobile robot path planning with goods inspection task

Figure 14. QR-code recognition results obtained by the robot during the automated goods-inspection task, the decoded information corresponds to product identity and location

3.5. Discussion

The experimental and simulation results demonstrate the effectiveness of the proposed intelligent navigation framework integrating image-based environment modeling, PSO global path planning, and DWA local control. The image-processing module successfully generated accurate occupancy maps, achieving an average localization error of less than 1 cm in both x and y directions. This level of precision is sufficient for most indoor warehouse applications, where position tolerances typically range from 1–2 cm. The CNN-based goal recognition system achieved an overall accuracy of 99%, confirming that the network is robust and well-generalized for character and symbol detection tasks. The integration of this vision-based recognition allows the robot to autonomously identify target points without relying on manual input. When evaluating localization, the fusion of odometry and ArUco marker detection significantly reduced accumulated drift compared to odometry alone. The resulting trajectory closely matched the ground-truth path, with positional errors maintained within 1–5 cm, validating the benefit of combining global visual information with local

motion data. In both simulation and real-world tests, the PSO–DWA hybrid planner provided smooth and collision-free motion. The robot successfully reached all designated points and executed QR-code inspections at each location.

In various state of art, several studies have explored mobile-robot navigation by combining classical planning and perception modules, but with different design goals and validation settings. Global navigation was addressed using an $A*$-based strategy and an additional learning component, and a success rate of 87.5% was reported after training with an extreme gradient boosting (XGBoost) approach [5]. TurtleBot4 navigation was supported by an RGB camera and visual-odometry processing, and the reported localization error reached a minimum root mean square error (RMSE) of 0.35 m [33]. These works demonstrate that practical navigation can be achieved by integrating planning with perception and learning, but the reported performance depends strongly on sensing quality and the specific environment assumptions. Other approaches have focused on improving local motion behaviors and map reliability in more complex scenes. An improved DWA method combined with Q-learning was used to strengthen navigation capability in both static and dynamic environments [34]. Dijkstra was applied for global path computation, and map performance in a dynamic environment was reported with an accuracy of about 90% [35].

Compared with these studies, the proposed system emphasizes an end-to-end integration of global planning and real-time local control under an overhead-vision setup. A global path is generated by PSO, while DWA is executed at each control step to enforce collision-aware motion in real time. In addition, goal recognition is supported by CNN-based marker detection, and localization drift is reduced by combining odometry with ArUco-based correction. The combined design enables reliable navigation from simulation to real-world operation, while keeping the online control loop lightweight through DWA execution during motion. Although PSO required 477–692 s for global path generation, it was executed offline on a known static map and therefore did not affect the real-time DWA control loop. However, this relatively long computation time limits the scalability of the current framework and makes it unsuitable for frequent global replanning in large or rapidly changing environments. A detailed comparison with related studies is provided in Table 9.

Table 9. Comparison with related navigation studies

Study	Global planning	Local control	Perception/localization	Environment
Waga *et al.* [5]	$A*$	Not specified in the summary	Learning-assisted component (XGBoost mentioned)	Indoor navigation
Singh *et al.* [33]	Not specified in the summary	Not specified in the summary	RGB camera + visual odometry	Indoor navigation (TurtleBot4)
Chang *et al.* [34]	Not specified in the summary	Improved DWA + Q-learning	Not specified in the summary	Static and dynamic environments
Liu *et al.* [35]	Dijkstra	Not specified in the summary	Dynamic-map related method	Dynamic environment
This work	PSO	DWA	Ceiling camera map + odometry + ArUco correction + CNN goal recognition	Indoor workspace

4. CONCLUSION

This study developed and validated an intelligent navigation framework for a two-wheeled differential-drive mobile robot operating in warehouse-like environments. The proposed system integrates an image-based environmental modeling method, global path planning using PSO, and local trajectory tracking through the DWA. The environment modeling approach, based on ceiling-camera image processing, achieved centimeter-level accuracy in obstacle detection and workspace mapping. The CNN-based recognition of numeric goal markers reached 99% accuracy, enabling reliable automatic goal identification. The hybrid localization method combining odometry and ArUco marker detection effectively mitigated cumulative drift, ensuring accurate and stable navigation. Experimental results demonstrated smooth, collision-free movement and successful completion of goods-inspection tasks using QR code recognition. These findings confirm that the proposed system provides a robust and efficient solution for indoor warehouse automation. Future work will focus on extending the system to larger-scale and dynamic environments, integrating real-time simultaneous localization and mapping (SLAM) techniques and 3D LiDAR sensing to enhance autonomy and adaptability. Despite the promising results, several practical limitations should be addressed. The current system relies on a single ceiling-mounted camera for top-view observation and occupancy-grid construction; consequently, potential occlusions, limited field of view, and calibration drift may affect mapping and localization precision. Furthermore, the image-processing pipeline assumes controlled lighting conditions, meaning that strong shadows or sudden illumination changes could degrade segmentation quality. Additionally, the experimental validation was primarily conducted with static obstacles on a flat indoor floor, which may not fully represent dynamic environments or uneven surfaces with wheel slip. In future work, framework could be extended by incorporating multi-camera systems to mitigate

occlusion, introducing dynamic obstacle tracking for online replanning, and integrating LiDAR-based SLAM to support larger-scale, unstructured environments while maintaining real-time performance.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Mr. Nguyễn Tấn Dũng and Mr. Phạm Văn Luôn, students of the Mechatronics Engineering Program (K47), School of Engineering and Technology, Can Tho University, for their active participation and valuable assistance during the mobile robot experiments. The authors would like to thank Vinh Long University of Technology Education for supporting this research.

FUNDING INFORMATION

There is no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Thanh Quang Duc Le	✓	✓	✓		✓	✓	✓	✓	✓				✓	✓
Trong Hieu Luu	✓	✓		✓	✓	✓				✓	✓	✓	✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

REFERENCES

[1] F. Wang, W. Sun, P. Yan, H. Wei, and H. Lu, "Research on path planning for robots with improved A* algorithm under bidirectional JPS strategy," *Applied Sciences*, vol. 14, no. 13, 2024, doi: 10.3390/app14135622.
[2] H. Huang, Y. Li, and Q. Bai, "An improved a star algorithm for wheeled robots path planning with jump points search and pruning method," *Complex Engineering Systems*, vol. 2, no. 3, 2022, doi: 10.20517/ces.2022.12.
[3] D. Xiang, H. Lin, J. Ouyang, and D. Huang, "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot," *Scientific Reports*, vol. 12, no. 1, 2022, doi: 10.1038/s41598-022-17684-0.
[4] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, "The EBS-A* algorithm: an improved A* algorithm for path planning," *PLoS ONE*, vol. 17, no. 2, 2022, doi: 10.1371/journal.pone.0263841.
[5] A. Waga, A. B.-Ichou, S. Benhlima, A. Bekri, and J. Abdouni, "Efficient autonomous navigation for mobile robots using machine learning," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 3, pp. 3061–3071, Sep. 2024, doi: 10.11591/ijai.v13.i3.pp3061-3071.
[6] Y. Hanli, L. Qiliang, and Z. Weizhen, "Robot obstacle avoidance based on improved artificial potential field method," in *10th International Conference on Control, Automation and Information Sciences, ICCAIS 2021*, Oct. 2021, pp. 769–775, doi: 10.1109/ICCAIS52680.2021.9624644.
[7] C. Cheng, Q. Sha, B. He, and G. Li, "Path planning and obstacle avoidance for AUV: a review," *Ocean Engineering*, vol. 235, Sep. 2021, doi: 10.1016/j.oceaneng.2021.109355.
[8] Z. Wu, J. Dai, B. Jiang, and H. R. Karimi, "Robot path planning based on artificial potential field with deterministic annealing," *ISA Transactions*, vol. 138, pp. 74–87, Jul. 2023, doi: 10.1016/j.isatra.2023.02.018.
[9] R. Szczepanski and F. Gul, "Local path planning algorithm based on artificial potential field with adaptive scaling factor," in *2024 28th International Conference on Methods and Models in Automation and Robotics*, Aug. 2024, pp. 229–234, doi: 10.1109/MMAR62187.2024.10680843.
[10] T. Xing, X. Wang, K. Ding, K. Ni, and Q. Zhou, "Improved artificial potential field algorithm assisted by multisource data for AUV path planning," *Sensors*, vol. 23, no. 15, Jul. 2023, doi: 10.3390/s23156680.
[11] W. Yang *et al.*, "Improved artificial potential field and dynamic window method for amphibious robot fish path planning," *Applied Sciences*, vol. 11, no. 5, pp. 1–15, 2021, doi: 10.3390/app11052114.
[12] J. Li, Y. Hu, and S. X. Yang, "A novel knowledge-based genetic algorithm for robot path planning in complex environments," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 2, pp. 375–389, 2025, doi: 10.1109/TEVC.2025.3534026.
[13] K. S. Suresh, R. Venkatesan, and S. Venugopal, "Mobile robot path planning using multi-objective genetic algorithm in industrial automation," *Soft Computing*, vol. 26, no. 15, pp. 7387–7400, 2022, doi: 10.1007/s00500-022-07300-8.
[14] Y. Li, J. Zhao, Z. Chen, G. Xiong, and S. Liu, "A robot path planning method based on improved genetic algorithm and improved dynamic window approach," *Sustainability*, vol. 15, no. 5, Mar. 2023, doi: 10.3390/su15054656.

[15] C. Liu, A. Liu, R. Wang, H. Zhao, and Z. Lu, "Path planning algorithm for multi-locomotion robot based on multi-objective genetic algorithm with elitist strategy," *Micromachines*, vol. 13, no. 4, 2022, doi: 10.3390/mi13040616.
[16] M. Ronghua, C. Xinhao, W. Zhengjia, and Du Xuan, "Improved ant colony optimization for safe path planning of AUV," *Heliyon*, vol. 10, no. 7, 2024, doi: 10.1016/j.heliyon.2024.e27753.
[17] S. Zhang, J. Pu, and Y. Si, "An adaptive improved ant colony system based on population information entropy for path planning of mobile robot," *IEEE Access*, vol. 9, pp. 24933–24945, 2021, doi: 10.1109/ACCESS.2021.3056651.
[18] L. Wu, X. Huang, J. Cui, C. Liu, and W. Xiao, "Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot," *Expert Systems with Applications*, vol. 215, 2023, doi: 10.1016/j.eswa.2022.119410.
[19] H. Wang, S. Wang, and T. Yu, "Path planning of inspection robot based on improved ant colony algorithm," *Applied Sciences*, vol. 14, no. 20, 2024, doi: 10.3390/app14209511.
[20] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Computers and Industrial Engineering*, vol. 156, 2021, doi: 10.1016/j.cie.2021.107230.
[21] Z. Yu, J. Mao, and W. Qi, "Path planning algorithm of mobile robot based on improved Q-learning," in *2024 7th International Conference on Advanced Algorithms and Control Engineering, ICAACE 2024*, 2024, pp. 1450–1453, doi: 10.1109/ICAACE61206.2024.10549135.
[22] E. S. Low, P. Ong, C. Y. Low, and R. Omar, "Modified Q-learning with distance metric and virtual target on path planning of mobile robot," *Expert Systems with Applications*, vol. 199, 2022, doi: 10.1016/j.eswa.2022.117191.
[23] A. Gharbi, "A dynamic reward-enhanced Q-learning approach for efficient path planning and obstacle avoidance in mobile robotics," *Applied Computing and Informatics*, vol. 22, no. 3-4, pp. 262–275, 2024, doi: 10.1108/ACI-10-2023-0089.
[24] H. Du, B. Hao, J. Zhao, J. Zhang, Q. Wang, and Q. Yuan, "A path planning approach for mobile robots using short and safe Q-learning," *PLoS ONE*, vol. 17, no. 9, 2022, doi: 10.1371/journal.pone.0275100.
[25] Q. Zhou, Y. Lian, J. Wu, M. Zhu, H. Wang, and J. Cao, "An optimized Q-learning algorithm for mobile robot local path planning," *Knowledge-Based Systems*, vol. 286, 2024, doi: 10.1016/j.knosys.2024.111400.
[26] T. Dam, G. Chalvatzaki, J. Peters, and J. Pajarinen, "Monte-carlo robot path planning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 11213–11220, 2022, doi: 10.1109/LRA.2022.3221234.
[27] T. Wang, L. Wang, D. Li, J. Cai, and Y. Wang, "Monte Carlo-based improved ant colony optimization for path planning of welding robot," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 7, 2023, doi: 10.1016/j.jksuci.2023.101603.
[28] A. Castellini, E. Marchesini, and A. Farinelli, "Partially observable Monte Carlo planning with state variable constraints for mobile robot navigation," *Engineering Applications of Artificial Intelligence*, vol. 104, 2021, doi: 10.1016/j.engappai.2021.104382.
[29] P. Xu *et al.*, "Contact sequence planning for hexapod robots in sparse foothold environment based on Monte-Carlo tree," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 826–833, 2022, doi: 10.1109/LRA.2021.3133610.
[30] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
[31] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, Apr. 1997, doi: 10.1109/100.580977.
[32] Z. Liu, H. Mao, C. Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11966–11976, doi: 10.1109/CVPR52688.2022.01167.
[33] W. A.-Neto, M. F. D. Santos, V. B. Schettino and P. Mercorelli, "Robotics as a mobile laboratory: evaluating educational platforms for industrial instrumentation using TurtleBot 4," *2026 27th International Carpathian Control Conference*, Szilvásvárad, Hungary, 2026, pp. 49-54, doi: 10.1109/ICCC71363.2026.11593252.
[34] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement-based mobile robot path planning with improved dynamic window approach in unknown environment," *Autonomous Robots*, vol. 45, no. 1, pp. 51–76, 2021, doi: 10.1007/s10514-020-09947-4.
[35] L. S. Liu *et al.*, "Path planning for smart car based on Dijkstra algorithm and dynamic window approach," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, 2021, doi: 10.1155/2021/8881684.

BIOGRAPHIES OF AUTHORS

Thanh Quang Duc Le received his M.Sc. degree in Electrical Engineering from Vinh Long University of Technology Education, Vietnam, in 2021, and his B.Eng. degree in Industrial Electrical Engineering from Ho Chi Minh City University of Technology and Education, Vietnam, in 2011. He is currently a lecturer and Ph.D. candidate in Electrical Engineering at Vinh Long University of Technology Education. His research interests include control engineering and mobile robotics. He can be contacted at email: ducltq@vlute.edu.vn.

Trong Hieu Luu received his B.E. degree in Mechatronic in 2010 and M.S. degree in Automation and Control in 2016 in Can Tho University. He received the Ph.D. degree in Course of Applied Marine and Environmental Studies from Tokyo University of Marine Science and Technology in 2020. He is currently a lecturer in mechatronics at the College of Engineering, Can Tho University. His research interests include computer vision, AI with image processing and robotic technology. He can be contacted at email: luutronghieu@ctu.edu.vn.